intel

APPLICATION NOTE

AP-96

July 1980

Designing iSBX™ MULTIMODULE™ Boards

Stephen Grubb
OEM Microcomputer Systems Applications

AFN-01931A

INTRODUCTION

Intel's single board computers and the MULTIBUS system bus have become de facto industry standards in the microcomputer board market. The speed and capability of the bus coupled with the functionality and performance of the boards have been used to solve a large number of problems. iSBC products are in applications ranging from simple single board relay replacement to sophisticated multi-board business systems supporting large hard disk files. However, even with the range of functionality provided by standard iSBCs and expansion boards, designers have felt the need to design custom MULTIBUS-compatible boards to fit their application. Until the introduction of the iSBX concept, these custom boards had to be implemented using a separate MULTIBUS form factor board.

Intel has recently introduced a new line of board products and a new bus which are destined to become another industry standard because of the niche they fill. The new iSBX MULTIMODULE boards are designed to extend the functional capabilities of single board computers at a much lower cost than previously possible. iSBX MULTIMODULE boards are supported by a new bus — the iSBX bus, which allows the MULTIMODULE boards to be added directly to the on-board microprocessor bus. iSBX MULTIMODULE boards are from 10 to 20 square inches in size, therefore permitting small modular increments to a single board computer's capabilities.

System designers now have the capabilities of using either standard iSBCs or iSBX MULTIMODULE boards, or designing custom MULTIBUS compatible or iSBX MULTIMODULE boards. Cost-effective solutions are easily realized because of this added flexibility.

This application note discusses the iSBX MULTIMODULE concept, currently available MULTIMODULE boards and the iSBCs which support these boards. The iSBX bus interface specifications are discussed next, followed by consideration for designing custom iSBX MULTIMODULE boards. A specific design example using an Intel® 8279 Programmable Keyboard/Display Controller is presented.

The objective of the note is to introduce the reader to the iSBX MULTIMODULE concept for expanding iSBC functionality and to illustrate how a designer can effectively use this concept with either standard or custom iSBX boards.

References to further documentation on the iSBX bus, specific iSBX MULTIMODULE boards and iSBC host boards currently available may be found in the Related Intel Publications section in the front overleaf of this application note.

iSBX™ MULTIMODULE™ BOARD CONCEPT

The iSBX MULTIMODULE board concept was developed to provide the users of Intel single board computers (iSBCs) with a convenient method to incrementally expand the I/O or the computing capabilities of a single board computer. This expansion is done through the use of a new interface called the iSBX bus interface. This interface gives the user the capability of adding I/O mapped functions directly onto the microprocessor bus via plug-in modules that connect to the iSBC board by means of a special iSBX connector. With the use of this new bus interface, it is now possible to expand or add new features to your iSBC system without incurring large costs and long engineering development times.

There are a number of unique advantages to using the iSBX bus interface for system expansion rather than adding a separate expansion board to your system. First, when expansion is required, the user needs only to buy what is required for the application. Second, it is now possible to return to one board solutions for small systems. One board solutions eliminate the need for expensive backplanes and cardcages. Next, the iSBX interface connects directly to the microprocessor or local bus, as opposed to interfacing to the MULTIBUS system bus, therefore I/O expansion does not require system bus cycles. To the CPU, the iSBX board looks like any other on-board I/O device (Figure 1). Address decode logic exists on the iSBC host board for each iSBX connector on the host board.

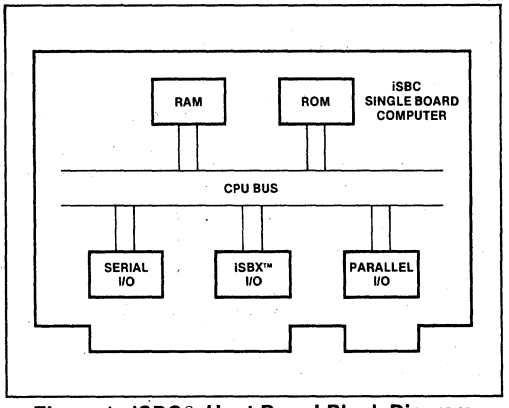

Figure 1. iSBC® Host Board Block Diagram

Third, if there is no iSBC or MULTIBUS compatible expansion board available to fit the needs of your application or if the expansion boards available offer more capability than required, then it is possible to design a custom iSBX MULTIMODULE board. Custom iSBX boards offer several advantages over custom MULTIBUS boards: they require less board real estate (10 or 20

square inches versus 81 square inches) and less engineering design time; consequently, they cost considerably less to implement. Additional capability is therefore achieved with maximum productivity.

Currently available Intel iSBX MULTIMODULE Boards include:

1) iSBX 350 Parallel I/O MULTIMODULE board which contains 24 programmable I/O lines with sockets for line drivers and terminators.
2) iSBX 351 Serial I/O MULTIMODULE board containing one RS232 or RS449/422 programmable synchronous/asynchronous communications channel and two timers.
3) iSBX 331 Fixed/Floating Point Math MULTIMODULE board which permits fixed or floating point mathematics via the Intel 8231 device.
4) iSBX 332 Floating Point Math MULTIMODULE board which permits floating point mathematics using the Intel and proposed IEEE floating point standards via an Intel 8232 device.

With these iSBX MULTIMODULE boards and other soon-to-be-announced boards, the capability now exists to economically tailor a single board computer to the application using off-the-shelf products.

iSBX™ MULTIMODULE™ SYSTEM INTERFACE

This section begins by describing the basic system elements used in an iSBX MULTIMODULE interface configuration and then defines the interface signals used for the communication between these elements. The specifications contained in this application note are included for descriptive and tutorial purposes only. The ultimate source for this information is the iSBX Bus Specification which is referenced in the front overleaf of this note.

Host Boards

The host board provides an electrical and mechanical interface for the iSBX expansion module. The host board is the master of the communications between the host and iSBX board, it controls the address and command signals.

A new generation of iSBX bus compatible host boards are evolving. The first board available from Intel is the iSBC 80/10B Single Board Computer. The 80/10B contains an 8080A CPU operating at 2 MHz, 1K bytes of RAM with sockets available for expansion to 4K bytes of RAM, sockets for up to 16K bytes of EPROM, 24 parallel I/O lines, a programmable synchronous/asynchronous communications interface and a fixed 1.04 msec timer. The 80/10B has one iSBX connector, permitting the use of an iSBX MULTIMODULE board.

The second iSBC board available supporting iSBX boards is the iSBC 80/24 Single Board Computer. The 80/24 board, which supports two iSBX MULTIMODULE boards, contains an 8085A-2 CPU operationg at 4.8 or 2.4 MHz, 4K bytes of RAM, sockets for up to 32K bytes of EPROM, 48 parallel I/O lines, a programmable synchronous/asynchronous communications interface, three programmable interval timers and a programmable interrupt controller. Further RAM expansion on the 80/24 board is accomplished by the addition of an iSBC 301 4K byte RAM MULTIMODULE board which expands the RAM by an additional 4K bytes for a total of 8K bytes. The iSBC 301 MULTIMODULE board is not iSBX bus compatible; it is attached via pins and sockets in the RAM section of the host board.

iSBX™ MULTIMODULE™ Boards

The iSBX MULTIMODULE boards communicate with the host boards via the iSBX bus interface. These iSBX boards are I/O mapped through pre-defined select lines to specific port addresses. The iSBX bus currently defines an 8-bit data path compatible with both 8 and 16-bit future iSBC host boards. Examples of possible iSBX expansion boards include a floppy disk controller, a cassette interface, analog-to-digital converter or digital-to-analog converter boards, an interface to the IEEE 488 Bus and a video graphics display interface board.

There are two standard sizes of iSBX boards: a single-wide board measuring 7.24 by 9.40 cm (2.85 by 3.70 inches) and a double-wide board measuring 7.24 by 19.05 cm (2.85 by 7.50 inches). The iSBX MULTIMODULE boards mount onto any microcomputer board containing an iSBX connector and mounting hole. The iSBX boards physically plug into the iSBX connector on the host board and are secured with a nylon stand-off and screws. The mounting hardware supplied as part of the iSBX board includes:

1) One nylon spacer, 1/2″ threaded
2) Two nylon screws, 1/4″ 6-32
3) One 36-pin connector, factory-installed onto the iSBX module. (These may also be purchased from Intel.)

The interconnection between the host board and iSBX board, as well as the mounting clearances, may be seen in Figures 2 and 3.

NOTE

The iSBX board, when installed onto a host board, occupies an additional card slot adjacent to the base board in an iSBC 604/614 Cardcage. However, the base board may be inserted in the top card slot of the cardcage. If this is done, no additional slots are required.

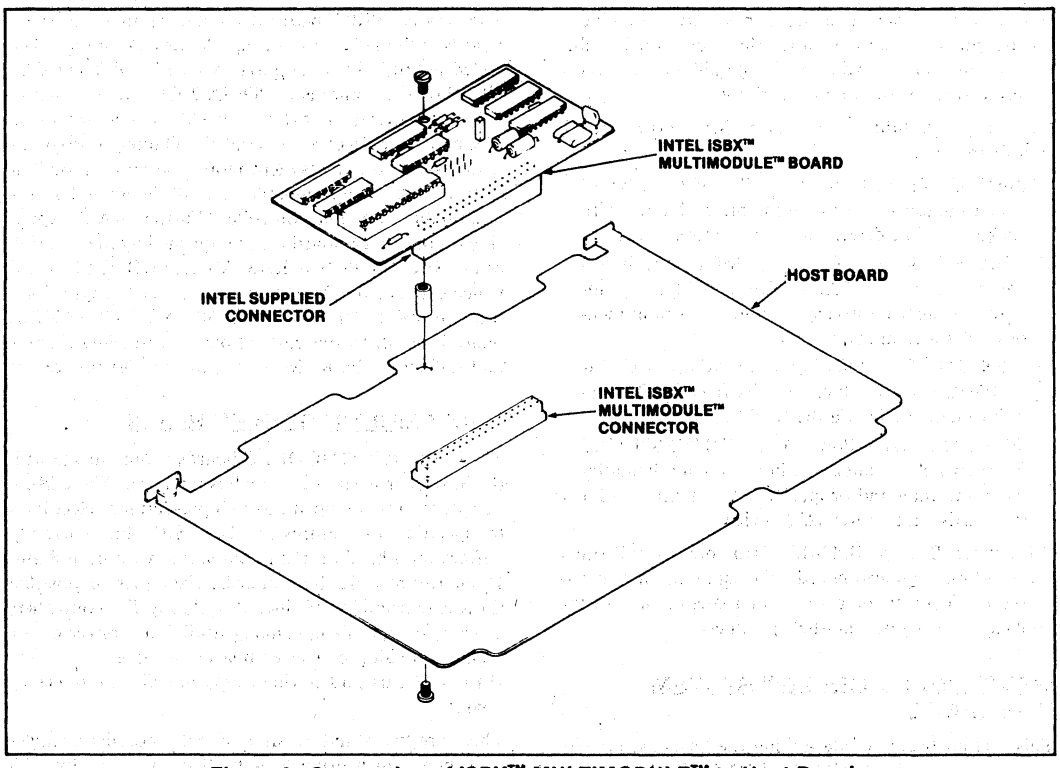

Figure 2. Connection of ISBX™ MULTIMODULE™ to Host Board

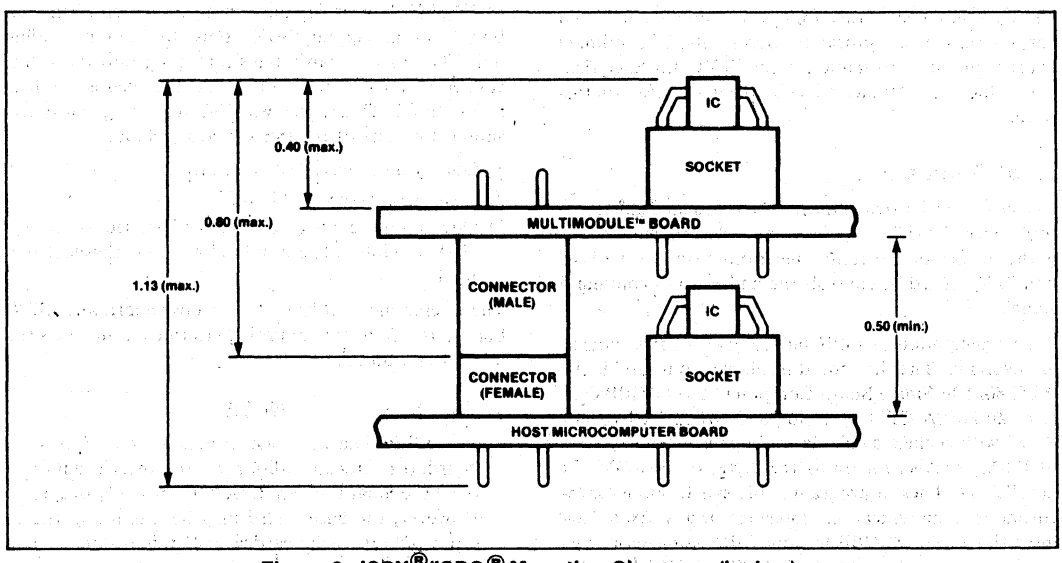

Figure 3. ISBX®/ISBC® Mounting Clearance (Inches)

iSBX™ Connector

The iSBX interface connector is a 36-pin custom made connector that was designed by Intel especially for this interface. The connector is plastic with gold plated contact pins for maximum reliability. The connector for the iSBX interface was designed for high reliability and durability. The connection between the host board and the iSBX MULTIMODULE board was extensively tested for vibration, shock, humidity, and temperature to insure that the connection is rugged enough to be used in severe environments. This connection was tested for the following environment:

Vibration: Sweeping from 10 Hz to 55 Hz and back to 10 Hz at a distance of 0.010 inches peak-to-peak, lasting 15 minutes in each of the three planes.

Shock: 30g's of force for an 11-msec duration, three times in three planes, both sides (total of 18 drops).

Humidity: 90% maximum relative (no condensation).

Temperature: 0 to 55°C (32-131°F) free moving air across the base board and the iSBX MULTIMODULE board.

Further information on the reliability testing that was done on this inter-connection, or reliability information on the iSBX MULTIMODULE boards in general, is contained in the Reliability Report, RR-29, "Intel iSBX MULTIMODULE Boards and iSBC 80/10B Single Board Computer," listed in the overleaf of this note.

The male half of this connector is available from Intel in the form of the iSBX 960-5 package which contains five of the connectors.

iSBX™ Bus Interface Signals

The iSBX bus interface signals are grouped into six basic groups, or classes, according to the functions performed relative to the interface:

These signals are:

CONTROL LINES
ADDRESS LINES
DATA LINES
INTERRUPT LINES
OPTIONAL LINES
POWER LINES

Many of the signals on the iSBX bus are active-low, meaning a low level on a control signal of the bus indicates a logic "1" value, while a low level on an address or data signal of the bus represents a logic "0" value.

NOTE

In this application note, an active-low signal will be designated by placing a slash (/) after the mnemonic for the signal.

Appendix A contains a pin assignment list of the following signals:

CONTROL LINES

The following signals are classified as control lines:

1) COMMANDS — IORD/, IOWRT/
2) DMA — DMRQT, MDACK/, TDMA
3) INITIALIZE — RESET
4) CLOCK — MCLK
5) SYSTEM CONTROL — MWAIT/, MPST/

Command Lines (I/O READ, I/O WRITE)

The command lines are active-low signals which control the communication link between the host board and the iSBX board. An active command line conditioned by chip select indicates to the iSBX board that the address lines are valid and the iSBX board should perform the specified operation.

DMA Lines (MDRQT, MDACK/, TDMA)

The DMA lines control the communication link between the DMA device on the host board and the iSBX module. DMRQT is an active-high output signal from the iSBX board to the host board's DMA device requesting a DMA cycle. MDACK/ is an active-low input signal to the iSBX board from the host board DMA device acknowledging that the requested DMA cycle has been granted. TDMA is used by the iSBC board to terminate DMA activity. The use of the DMA lines is optional as not all host boards will provide DMA channels nor will all iSBX boards be capable of supporting them.

Initialize Line (RESET)

This active-high input line to the iSBX board is generated by the host board to put the iSBX board into a known internal state.

Clock Line (MCLK)

This input line to the iSBX board is a timing signal. The clock frequency is 10 MHz (+0%, −10%), and the clock is asynchronous with respect to all other iSBX bus signals.

System Control Lines (MWAIT/, MPST)

These output signals from the iSBX board control the state of the system. Active MWAIT/ (active-low) will

put the CPU on the host board into a wait state, providing additional time for the iSBX board to perform the requested operation. MPST/ is an active-low signal (usually tied to signal ground) that informs the host board I/O decode logic that an iSBX module has been installed.

ADDRESS AND CHIP SELECT LINES

The address and chip select lines are made up of the following signals:

1) ADDRESS LINES — MA0, MA1, MA2
2) CHIP SELECT LINES — MCS0/, MCS1/

Address Lines (MA0, MA1, MA2)

These active-high input lines to the iSBX boards are generally the least three significant bits of the I/O addresses. In conjunction with the command and chip select lines, they establish the I/O port address being accessed.

Chip Select Lines (MCS0/, MCS1/)

These active-low input lines to the iSBX board are the result of the host board I/O decode logic. When active, the MCS/ lines condition the I/O command signals and thus enable communication between the iSBX board and the host board.

DATA LINES (MD0-MD7)

There are eight bidirectional data lines. These active-high lines are used to transmit or receive information to or from the iSBX ports. MD0 is the least significant bit.

INTERRUPT LINES (MINTR0, MINTR1)

These active-high output lines from the iSBX board are used to make interrupt requests to the host board. These lines are jumper enabled and disabled on the host board via wire wrap posts.

OPTION LINES (OPT0, OPT1)

These two signals are reserved lines that are connected to wire wrap posts on both the host board and the iSBX MULTIMODULE board. They are for unique requirements where a user needs a host board or MULTIBUS bus signal on the iSBX module.

POWER LINES

All host boards provide +5 volts as well as ±12 volts to the iSBX MULTIMODULE board along with signal ground. All power supply voltages are ±5%. Table 1 gives the power supply specifications for the iSBX interface.

Table 1. Power Supply Specifications

Minimum (volts)	Nominal (volts)	Maximum (volts)	Maximum (current)*
+4.75	+5.0	+5.25	3.0A
+11.4	+12	+12.6	1.0A
−12.6	−12	−11.4	1.0A
—	GND	—	6.0A

*Per iSBX MULTIMODULE board mounted on base board.

iSBX™ BUS INTERFACING

This section of the application note focuses on the iSBX interface and design considerations related to interfacing with the iSBX bus. It discusses the way the major operations like READ, WRITE, and DMA work, and the timing diagrams associated with each. There is also a discussion on other considerations for designing with the iSBX bus.

Bus Timing

The AC timing specifications for the iSBX bus interface can be found in Appendix B of this application note. It should be emphasized that the interface timing between the host board and the iSBX MULTIMODULE board is very critical. This is largely due to the fact that the iSBX board is attached directly to the microprocessor bus. If the timing specifications are not met, unpredictable and possibly intermittent operation of the host board may result.

Command Operations

The command lines (IORD/, IOWRT) are driven from the host board by three-state drivers with pull-up resistors or standard TTL totem-pole drivers. These lines indicate to the iSBX board that action is being requested. There are two types of operations for each command line and it is the iSBX board that determines which operation is to be performed.

READ OPERATIONS (IORD/)

Two different types of read operations are possible. The first type of read is called a full speed I/O READ. The host board generates a valid I/O address (MA0-MA2) and a valid chip select signal (MCS1/) which is then sent to the iSBX board; after the set-up times are met, the host board activates the IORD/ line. At this time, the iSBX board must generate valid data from the addressed I/O port in less than 250 ns. The host board then reads the data and removes the READ command, address and chip selects. These are shown in the timing diagram for this operation (Figure 4). The second type of read operation is called an I/O READ with Wait. This READ is used by iSBX boards that cannot perform a full speed read operation. Under this operation the

host board generates the valid address and chip select signals, as in the full speed read. But this time the iSBX board will activate the MWAIT/ signal, which in turn removes the READY input to the CPU, putting it into a Wait state. The CPU, however, first activates the IORD/ signal before going into the Wait state. After valid data is placed on the iSBX data bus by the iSBX board, the iSBX board will remove the MWAIT/ signal. The host board will then read the data and remove the command, address, and chip select lines. This I/O READ with Wait operation is shown in Figure 5.

WRITE OPERATIONS (IOWRT/)

There are also two types of write operations possible: the type performed is again determined by the iSBX board. In the full speed I/O WRITE operation, the host board generates a valid I/O address and chip select and then activates the IOWRT/ line after the necessary set-up times are met. The IOWRT/ line, after being activated, will remain active for 300 ns and the data will be valid for 250 ns before the IOWRT/ command is removed. The host board will then remove the data, address, and chip select lines after the hold times are met, as shown in the timing diagram of this operation (Figure 6).

This second write operation is the I/O WRITE with Wait operation. This WRITE is used by the iSBX boards that cannot write into an I/O port with the full speed write specifications. The host board again generates valid address and chip select signals as in the full speed write operation. However, this time the iSBX board generates the MWAIT/ signal based on address information (chip select and MA0-MA1). The activation of MWAIT/ causes the removal of READY to the CPU, thus causing the CPU to go into a Wait state. The iSBX board removes the MWAIT/ signal (allowing the CPU to leave its Wait state) when it has satisfied the WRITE pulse width requirements. At this time the board removes the WRITE command, followed by the data, address, and chip select lines. This I/O WRITE with Wait operation can be seen in Figure 7.

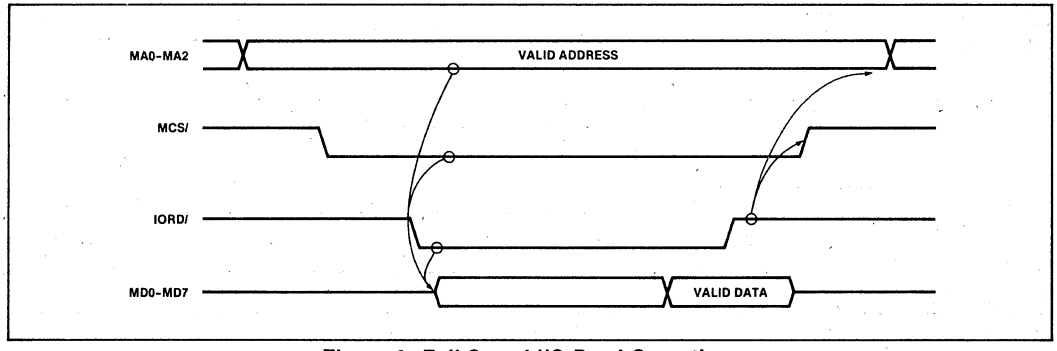

Figure 4. Full Speed I/O Read Operation

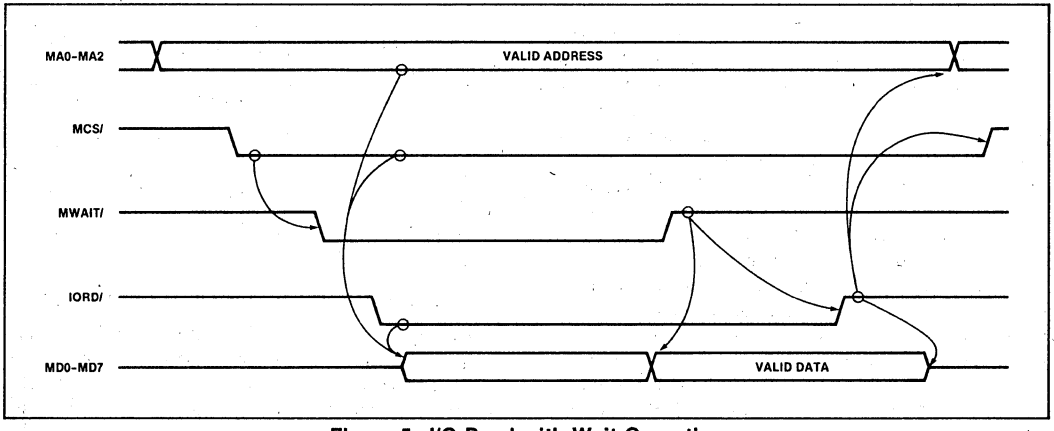

Figure 5. I/O Read with Wait Operation

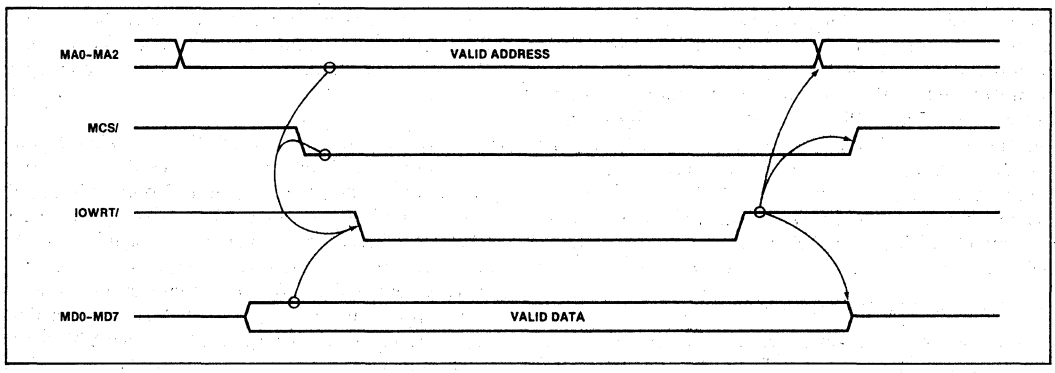

Figure 6. Full Speed I/O Write Operation

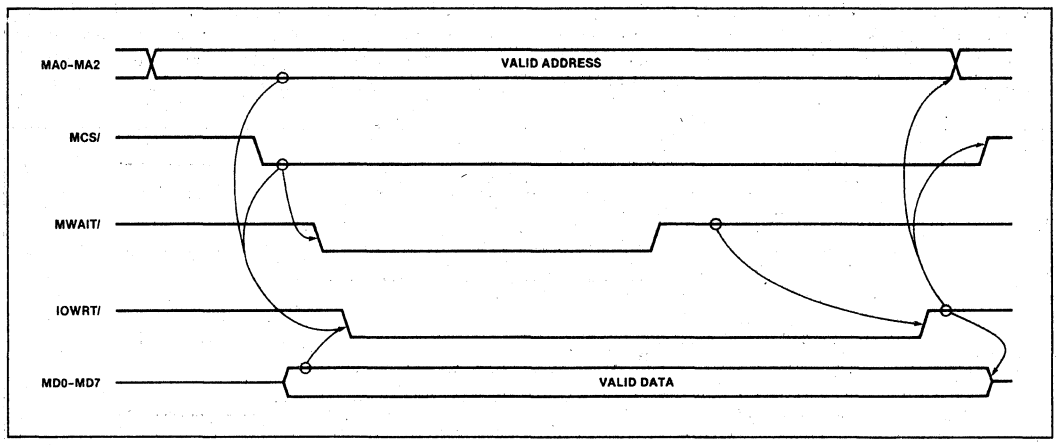

Figure 7. I/O Write with Wait Operation

iSBX™ Addressing

The iSBX boards are addressed by the host board through the use of the address lines MA0, MA1 and MA2, and the chip select lines MCS0/ and MCS1/. The host board decodes the I/O addresses and in turn generates the chip selects for the iSBX boards. In an 8-bit system the host board decodes the high order 13 address bits and generates the appropriate chip select corresponding to those address bits. The low order three address bits are passed to the iSBX board via MA0-MA2. Thus, a host board reserves two blocks of eight I/O ports for each iSBX connector. There can be as many as three iSBX connectors per host board, therefore a total of 48 addresses or six blocks of eight I/O ports that can be reserved for the iSBX boards. Table 2 contains a list of the I/O addresses and their corresponding host board iSBX port assignments of the iSBC 80/10B and iSBC 80/24 host boards.

Table 2. iSBX™ Host Board Port Assignment

iSBX™ Connector Number	Chip Select	iSBX™ Port Addresses
iSBC 80/10B Connector	MCS0/ MCS1/	F0-F7 F8-FF
iSBC 80/24 First Connector	MCS0/ MCS1/	F0-F7 F8-FF
iSBC 80/24 Second Connector	MCS0/ MCS1/	C0-C7 C8-CF

Considerations for iSBX™ Bus Interfacing

When designing with the iSBX interface it is important to note that the iSBX bus is not buffered on the host board. Since there is no isolation between the iSBX board and the host board CPU bus, a short between signal lines and power or ground could have a direct effect

on the CPU or the drivers and receivers associated with the CPU on the host board. This must be taken into consideration, especially when designing and debugging any custom designed iSBX MULTIMODULE board. It is usually during the development states of a product that these types of problems occur. One advantage to not buffering the iSBX bus is increased speed of data and command transfers. Applications requiring buffering may add the buffers on the iSBX board. A second advantage to not buffering is the saving of parts costs, board real estate and development time for the host board. Another consideration when designing with the iSBX interface is, if the application to be designed requires high throughput, like a floppy disk controller or a CRT controller, the designer may consider putting some type intelligent control of buffer RAM onto the iSBX board. By doing this, the transfer information can be stored in this buffer and the throughput of the system increased.

iSBX™ BUS LOADING REQUIREMENTS

Loading requirements for the iSBX bus have been broken up into two basic categories, output specifications and input specifications, which can be viewed in Tables 3 and 4. The output specifications are the requirements on the output drivers of the iSBX board and are the minimum drive requirements necessary. A good example of this would be that the data bus output drivers must be able to sink a minimum of 1.6 mA and maintain V_{OL} at a maximum of 0.5 volts and a minimum source of 200 μA, while providing a minimum output of 2.4 volts. The input specifications are the requirements on the receivers of the iSBX board. An example of this would be that the loading of the address lines (MA0- MA2) can be no greater than 0.5 mA with a minimum low threshold of 0.8 volts.

Optional Interface Lines

The iSBX interface has two optional lines which were included for the user to configure the iSBX board for special application needs. These two lines can be used in a number of ways helpful in unique situations. For example, they could be used as a way to get two extra interrupt lines down to the host board, thus yielding a total of four interrupt lines running between the iSBX MULTIMODULE board and the host board. They could also be used to get extra address lines, or even another clock signal to the iSBX board. They could also

Table 3. Output Specifications

Bus Signal Name	Type[2] Drive	I_{OL} Max – Min (mA)	@ Volts (V_{OL} Max)	I_{OH} Max – Min (μA)	@ Volts (V_{OH} Min)	C_O Min (pF)
MD0-MD7	TRI	1.6	0.5	–200	2.4	130
MINTR0-1	TTL	2.0	0.5	–100	2.4	40
MDRQT	TTL	1.6	0.5	–50	2.4	40
MWAIT/	TTL	1.6	0.5	–50	2.4	40
OPT1-2	TTL	1.6	0.5	–50	2.4	40
MPST/	TTL	Note 3				

Table 4. Input Specifications

Bus Signal Name	Type[2] Receiver	I_{IL} Max (mA)	@ Volts (V_{IN} Max)	I_{IH} Max (μA)	@ Volts (V_{IN} Min)	C_I Max (pF)
MD0-MD7	TRI	–0.5	0.4	70	2.4	40
MA0-MA2	TTL	–0.5	0.4	70	2.4	40
MCS0/-MCS1/	TTL	–4.0	0.4	100	2.4	40
MRESET	TTL	–2.1	0.4	100	2.4	40
MDACK/	TTL	–1.0	0.4	100	2.4	40
IORD/ IOWRT/	TTL	–1.0	0.4	100	2.4	40
MCLK	TTL	2.4	0.4	100	2.4	40
OPT1-OPT2	TTL	2.0	0.4	100	2.4	40

NOTES:
1. Per iSBX MULTIMODULE board.
2. TTL = standard totem-pole output. TRI = three-state.
3. iSBX MULTIMODULE board must connect this signal to ground.

be used to send a special status line to or from the iSBX MULTIMODULE board.

iSBX™ MULTIMODULE™ DESIGN EXAMPLE

This section covers the description of a custom iSBX MULTIMODULE board which uses the Intel 8279 Programmable Keyboard/Display Controller. This iSBX board, when added to an iSBC host board, provides an interface to a keyboard and display. A description of the hardware design considerations for breadboarding the hardware is presented. Following this, a software exerciser, useful for debugging the board, is described. A listing for the exerciser is contained in Appendix C.

Since the iSBX MULTIMODULE board was designed using the Intel 8279 Programmable Keyboard/Display Controller, a brief description of the 8279 is presented. The 8279 is a general purpose programmable keyboard and display I/O controller which was designed for use with the Intel microprocessors. The keyboard portion of this device is capable of providing a scanned interface to a 64-contact key matrix. It is also possible to interface to an array of sensors or a strobed keyboard, such as those of the Hall Effect or the ferrite variety. The 8279 provides a variety of keyboard inputs (i.e., 2-key lockout and N-key rollover), and all key entries are debounced and strobed into an 8-character FIFO. The display portion provides the user with a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used, as well as simple indicators. The 8279 is used in this iSBX design example to provide an interface of 2-key lockout with key debounce to a 64-character keyboard, and an interface for a 16-character, 18-segment alphanumeric display.

iSBX™ MULTIMODULE™ Board Design

The iSBX board that was designed for this application note contains a total of three IC's, the keyboard/display controller, a flip-flop, and a 3-to-8-line decoder. Figure 8 contains a block diagram of the hardware used in this design example. Figure 9 contains a schematic for the portion of the design example resident on the custom iSBX board.

The design offers the user some flexibility as to the type of display or keyboard to be attached. For example, if the application design was defined to be for a 7-segment, 16-character display (as the 8279 is designed to drive), a 4-to-16-line decoder along with the display drivers could be added to the iSBX board. Another idea would be to include everything except the display drivers and the display on the iSBX board, and to put the dis-

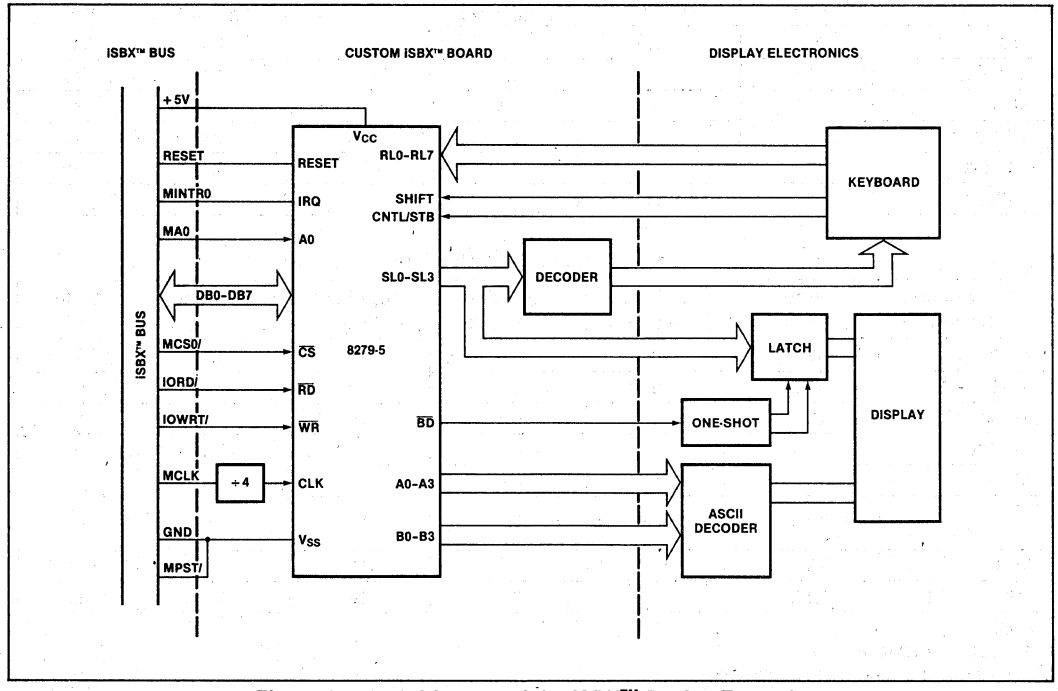

Figure 8. Block Diagram of the iSBX™ Design Example

play and drivers in with the keyboard. It is possible, and probably desirable in some applications, to incorporate some of the display electronics onto the iSBX MULTIMODULE board. Some of the IC's found in the display portion of this design could also have been placed on the iSBX board, as there is enough room on the finished product for doing so.

The design was very easy to implement because, with the exception of one signal, all of the iSBX bus signals necessary to drive the 8279 are connected directly without any extra logic needed. The one signal that would not connect directly to the interface is the clock signal MCLK from the bus to CLK on the controller. It is not possible to connect these two together as MCLK is a 10 MHz signal and the 8279 requires a maximum clock signal of 3.1 MHz to generate its internal timings. It is necessary to add a 74LS74 dual D-type flip-flop to divide the MCLK signal by 4 for the controller. With this exception, all other signals, DB0–DB7 to MD0–MD7, A_0 to MA0, CS/ to MCS0/, etc., are connected directly to the iSBX interface. To meet the timing requirements of the iSBX bus, a high speed version of the 8279, the 8279-5, is used.

The keyboard interface side of the iSBX board consists of a 3-to-8-line decoder, which is used for scanning the keyboard matrix. The 8279 scan lines SL0–SL2 are decoded by a 74LS156 open-collector output decoder and sent to the keyboard via a connector.

The display interface of the iSBX board consists of sending the scan lines and the display outputs to the display module via a connector. The scan lines SL0–SL3 are sent to the display drivers, and the display outputs A0–A3 and B0–B3 are sent to an ASCII to 18-segment decoder driver. The display is discussed in further detail in the next section of this application note.

Display Module Design

The display module design (Figure 10) consists of two 8-digit HDSP 6805 Alphanumeric Displays by Hewlett

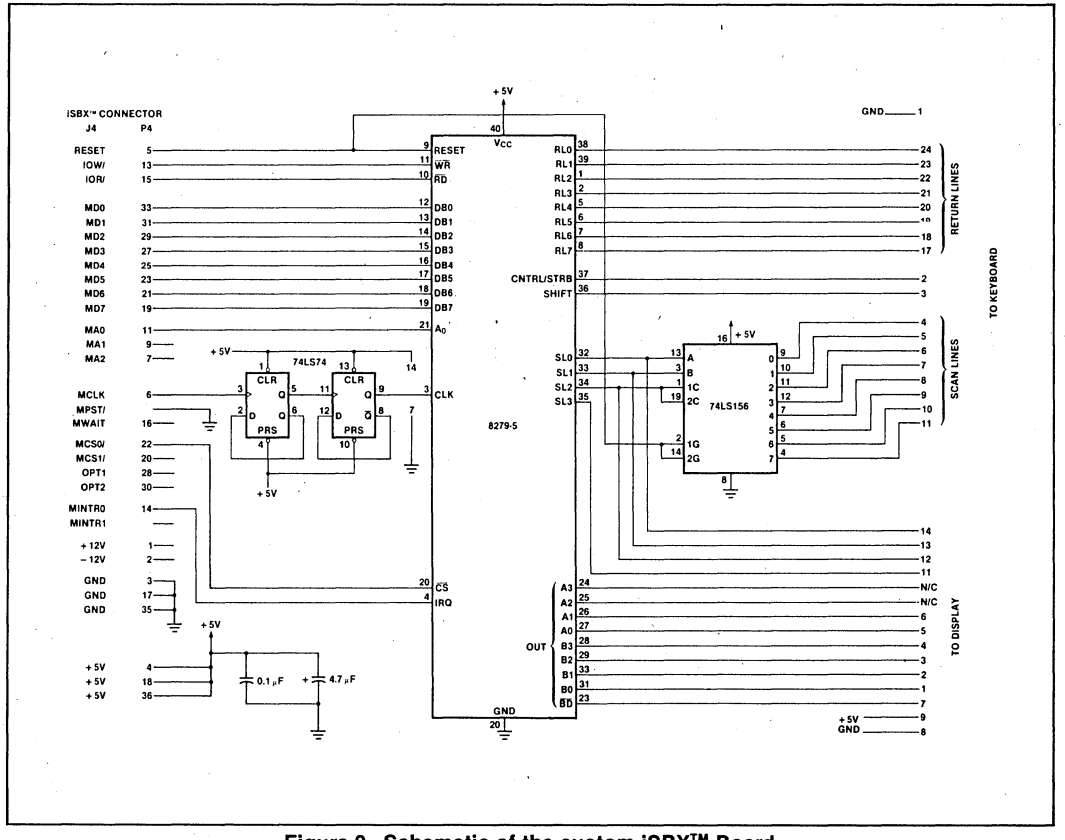

Figure 9. Schematic of the custom iSBX™ Board

Packard, the AC5947 ASCII to 18-segment decoder driver by Texas Instruments, two Signetics NE590 Peripheral Drivers, and a 74LS122 monostable multivibrator. The display is scanned by the outputs A0–A1 and B0–B3, which are connected to the inputs of the AC5947, and the SL0–SL3 outputs which are connected to the NE590 digit scanning circuitry. The interdigit blanking is provided by the 74LS122, which prevents a display ghosting type effect. With the 8279 display controller it is possible for the display to have either left entry, where the data enters from left to right across the display, overflowing in the left most display position, or right entry, where the data enters from the right side of the display and all previous data shifts left. Left entry was chosen for this example. The controller also provides commands for blanking or clearing the display.

Keyboard Interface Design

The eight output lines from the decoder on the iSBX board select 1-of-8 keyboard matrix rows for testing by the controller to see if a key depression has been made in the selected row. The keyboard matrix column output lines are connected directly to the return lines of the 8279, RL0–RL7. Open-collector outputs presented by individual keys within the matrix eliminate the need for isolation diodes when two keys in a given column are depressed. The keyboard/display controller has the option of using either scan keyboard, scan sensor matrix, or strobed input as modes of operation. With the scan keyboard mode there is a choice of using either 2-key lockout or N-key rollover for keyboard entry. The scan keyboard with 2-key lockout mode is used for this ex-

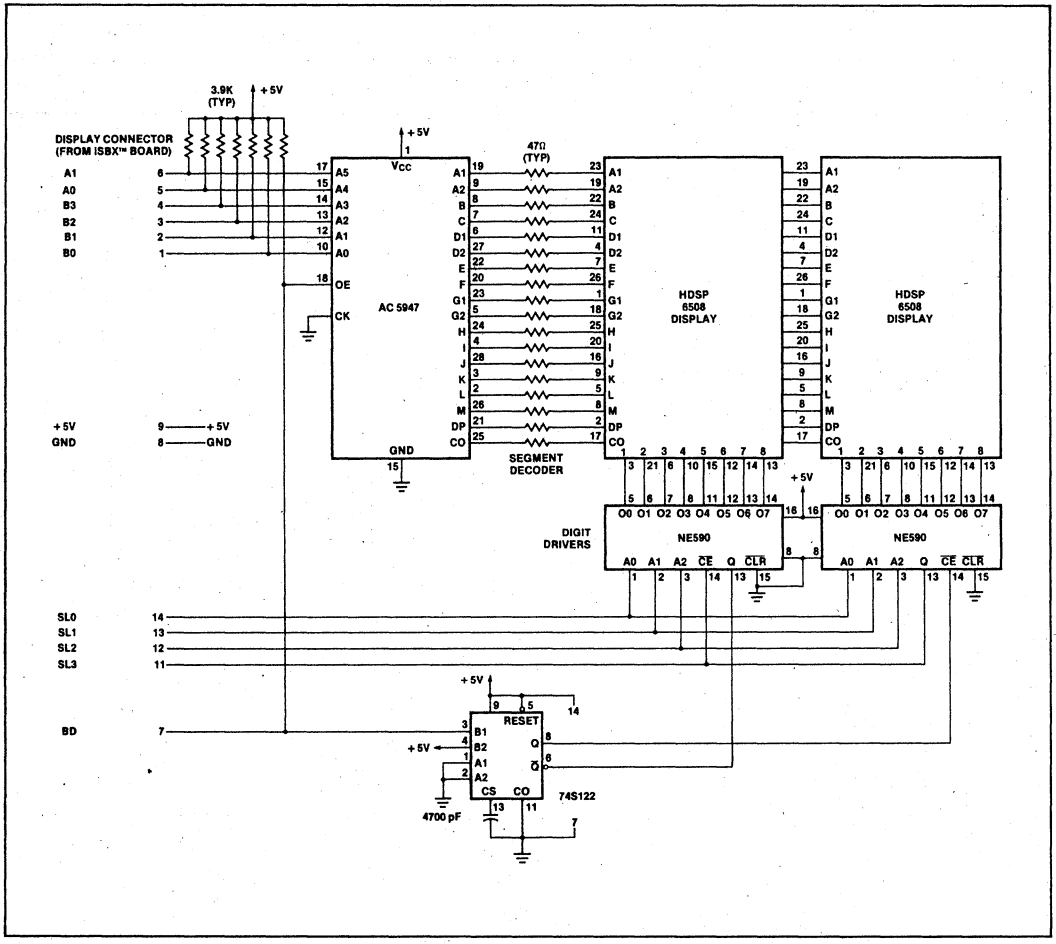

Figure 10. Display Module Schematic

ample. A diagram of the keyboard interfaces and matrix can be seen in Figure 11.

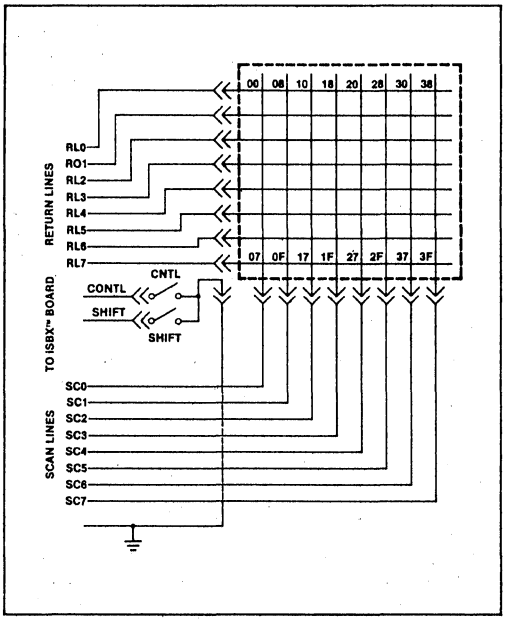

Figure 11. Keyboard Matrix Schematic

Operation with the iSBC®80/10B Single Board Computer

The 8279 on the iSBX expansion board is initialized to its mode of operation following a system reset. The keyboard mode of operation is to scan the keyboard with 2-key lockout, and the display mode is set for the 16-character left entry mode of operation. Upon receiving a character from the keyboard, the 8279 generates an interrupt along the MINTR0 line of the iSBX bus to the CPU. At this time the iSBC 80/10B board commences I/O read operations to the iSBX board by generating valid I/O address and chip select commands on the MA0 and MCS0/ signal lines. After the setup times are met, the 80/10B issues an I/O read command by asserting the IORD/ line on the bus, and the base board reads the data from the iSBX board and removes the IORD/, MA0, and MCS0/ signals from the bus. After the data has been read in from the keyboard, it must be output to the display. The iSBC 80/10B board starts an I/O write operation by generating a valid I/O address and the chip select signal with the MA0 and MCS0/ lines. After the valid setup times are met, the IOWRT/ line is activated by the base board. When the data has been valid for a minimum of 250 ns, the host board removes the IOWRT/ line. When the hold times have been met, the data, address and chip select lines are also removed. Figure 12 shows the timing diagrams just discussed.

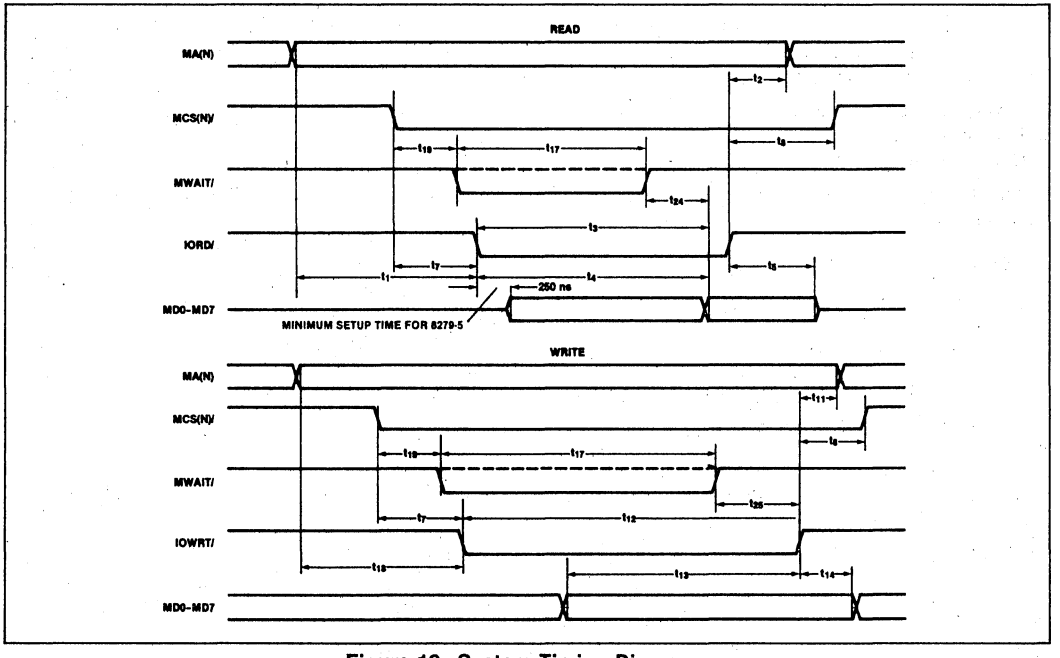

Figure 12. System Timing Diagrams

Breadboarding the Design

When doing the layout of the breadboard, it is also necessary to take into consideration the space required by the mounting holes and to plan the positioning of the components accordingly. (This information is available in the iSBX Bus Specification Manual.)

When attaching the breadboarded design, which typically contains raised wirewrap posts, it is necessary to raise the breadboard well above the host board. This can be accomplished by building a small cable and putting the breadboard on longer nylon standoffs. It is not recommended that the cable be longer than 15 cm (6 in.), otherwise bus timing problems could result.

With the breadboarding finished it is a good idea to recheck all wiring connections for possible errors. Also check all signal lines with an ohmmeter between power, and then ground, for potential shorts. An error at this point can cause serious damage to the host board!

Software Considerations

The software written for this application is an exerciser that is used for hardware checkout. It is a small program designed to echo characters from the keyboard to the display. The software was edited, assembled, linked and located with an Intel development system; it was then debugged with an in-circuit emulator. Both the software and the hardware debug is covered in the next section of this application note.

To facilitate this discussion the software exerciser is divided into three sections based upon the functions performed. The three functions are:

1) Keyboard interrupt routine
2) Initialization and flag checking routine
3) Character output routine

A complete listing of the software exerciser can be found in Appendix C.

KEYBOARD INTERRUPT ROUTINE

The 8279 generates an interrupt to the CPU whenever data is introduced into its FIFO/Sensor RAM. The interrupt is cleared by doing a data read. Whenever a key on the keyboard is depressed an interrupt is generated. Two things are required when an interrupt occurs. First, the keyboard input data must be retrieved and stored. Second, the interrupt routine must indicate that there is some data ready to be output to the display. Therefore, a buffer is created in memory (called "BUFF") at location 3C00H to store the keyboard data. A data present flag is set in a register (REG. C) to indicate that data is ready to be output and can be found in the buffer. In this way the interrupt routine is used to input characters from the keyboard to the input buffer. The buffer is then read by the output routine, which sends the characters to the display.

INITIALIZATION AND FLAG CHECKING ROUTINE

The initialization and flag checking routine first sets the stack pointer to the top of memory. After this the program proceeds to initialize the 8279 Keyboard/Display Controller to its proper mode of operation. The modes of operation used for this application note is scanned keyboard with 2-key lockout for the keyboard, and 16 characters with left entry for the display. As the 8279 has a desired internal operating frequency of 100 kHz, the frequency divider chain is programmed to divide by 19 hex, or 25 decimal. After the 8279 has been initialized, the program begins its next procedure of clearing the buffers. The keyboard input buffer, "BUFF", as well as the display buffer, "DBUFF", are both cleared to a blank display. This is done so that at the time of power up, the display will come up blank. With the initialization now complete, the program disables the interrupts and checks the data present flag for an indication that data might be present for output. If the data present flag is set, the output character routine is called; if it is not set, the interrupts are enabled and the program loops back around to check again. In summary, this routine initializes the 8279 and clears the buffers, and then loops on the data present flag looking for an indication that data is present in the input buffer. The input buffer is a one-byte wide buffer named "BUFF."

CHARACTER OUTPUT ROUTINE

The character output routine brings the character in from "BUFF" (the keyboard input buffer) and compares it to the characters located in a table. If the character can be matched to a character in the table it is replaced in "BUFF" with the corresponding character located in the same position of a second table. If there is no match, it is compared to the code for a control character. If there is no match with a control character, a compare is made to see if the character is a delete charater. When a match is found and the acceptable character is placed in "BUFF", the output routine shifts the data in the display buffer (Figure 13) one position to the left and places the character from the input buffer into the display buffer at position "DBUFF" + 15. Now that

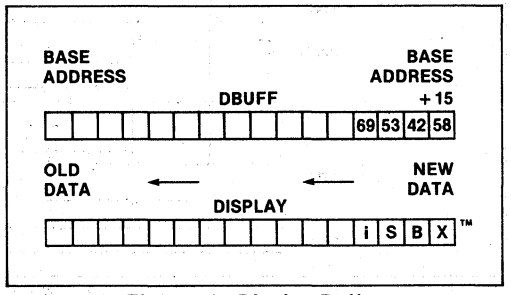

Figure 13. Display Buffer

the new information is in the display, the routine copies the complete contents of the display buffer, "DBUFF", to "DBUFF" + 15 to the display. In the case of the input character being matched up with a delete character, all information in the display buffer is shifted to the right one position and the ASCII code for a blank character is placed into the left-most position or the base address of "DBUFF", thus making the next character sent to the display a blank character. In the case of a control character, nothing is done and the program returns to the flag checking routine.

Debug Considerations

Hardware and software debug was accomplished using an iSBC 80/10B Single Board Computer, an iSBC 655 Chassis, an Intellec® Series II Model 230 Microcomputer Development System, and an ICE-80™ In-Circuit Emulator.

The software was down-loaded from the disk to the iSBC 80/10B board using the in-circuit emulator. The ICE™ module gives the engineer the capability of interrogating the iSBC system by allowing the user to access and display the CPU register contents, status, system memory contents, and all I/O devices and their data.

The iSBC 80/10B board was configured to enable interrupts from the iSBX board via the interrupt 0 line (MINTR0), which is connected to the interrupt pin of the 8080 CPU. The iSBX board was attached to the iSBC 80/10B board via the iSBX connector. The iSBC 80/10B board was powered-up and the iSBX board was checked for proper power and ground connections. The ICE-80 emulator was connected to the iSBC 80/10B board. Using the interrogation mode of the emulator, it is possible to check proper functioning of the iSBX board by sending and receiving data to/from the 8279. The keyboard can be tested by depressing a key on the keyboard and then examining the FIFO/Sensor RAM to see if the data was entered. The display RAM can also be read and written to for testing the interface to the display.

After this initial checking of the iSBX board, the software exerciser can then be down-loaded with the ICE module to further check the board.

SUMMARY

The objective of this application note is to introduce the reader to the iSBX MULTIMODULE concept for expanding a single board computer's functionality, and to illustrate how a designer can use this concept with either standard or custom iSBX boards. In contrast to system expansion using MULTIBUS-compatible boards, iSBX MULTIMODULE boards provide smaller, lower cost, incremental expansion. This application note explains how a custom iSBX board can be designed and debugged. Using this capability, it is now possible to more quickly add new VLSI technology to systems as the technology becomes available. Intel will continue to provide new iSBX MULTIMODULE boards and, because of the the publication of the iSBX Bus Specification and this application note, it will be easier for Intel's customers to also design and build their own custom iSBX boards.

APPENDIX A 9-52
APPENDIX B 9-53
APPENDIX C 9-55

APPENDIX A
iSBX™ SIGNAL PIN ASSIGNMENTS

Pin	Mnemonic	Description	Pin	Mnemonic	Description
35	GND	Signal Ground	36	+5V	+5 Volts
33	MD0	MDATA Bit 0	34	MDRQT	M DMA Request
31	MD1	MDATA Bit 1	32	MDACK/	M DMA Acknowledge
29	MD2	MDATA Bit 2	30	OPT0	Option 0
27	MD3	MDATA Bit 3	28	OPT1	Option 1
25	MD4	MDATA Bit 4	26	TDMA	Terminate DMA
23	MD5	MDATA Bit 5	24		Reserved
21	MD6	MDATA Bit 6	22	MCS0/	M Chip Select 0
19	MD7	MDATA Bit 7	20	MCS1/	M Chip Select 1
17	GND	Signal Ground	18	+5V	+5 Volts
15	IORD/	I/O Read Command	16	MWAIT/	M Wait
13	IOWRT/	I/O Write Command	14	MINTR0	M Interrupt 0
11	MA0	M Address 0	12	MINTR1	M Interrupt 1
9	MA1	M Address 1	10		Reserved
7	MA2	M Address 2	8	MPST/	iSBX MULTIMODULE Board Present
5	RESET	Reset	6	MCLK	M Clock
3	GND	Signal Ground	4	+5V	+5 Volts
1	+12V	+12 Volts	2	−12V	−12 Volts

All undefined pins are reserved for future use.

APPENDIX B
iSBX™ MULTIMODULE™ BOARD I/O AC SPECIFICATIONS

Symbol*	Parameter	Min (ns)	Max (ns)
t_1	Address stable before read	50	—
t_2	Address stable after read	30	—
t_3	Read pulse width	300	—
t_4(2)	Data valid from read	0	250
t_5(2)	Data float after read	0	150
t_6	Time between RD and/or WRT	—	Note 3
t_7	CS stable before CMD	25	—
t_8	CS stable after CMD	30	—
t_9	Power up reset pulse width	50 ms	—
t_{10}	Address stable before WRT	50	—
t_{11}	Address stable after WRT	30	—
t_{12}(2)	Write pulse width	300	—
t_{13}(2)	Data valid to write	250	—
t_{14}	Data valid after write	30	—
t_{15}	MCLK cycle	100	110
t_{16}	MCLK width	35	65
t_{17}(1)	MWAIT/ pulse width	0	4 ms
t_{18}	Reset pulse width	50 ms	—
t_{19}	MCS/ to MWAIT/ valid	0	75
t_{20}	DACK set up to I/O CMD	100	—
t_{21}	DACK hold	30	—
t_{22}	CMD to DMA RQT removed to end of DMA cycle	—	200
t_{23}	TDMA pulse width	500	—
t_{24}(1)	MWAIT/ to valid read data	—	0
t_{25}(1)	MWAIT/ to WRT CMD	0	—

NOTES:

1. Required only if WAIT is activated.
2. If MWAIT/ not activated.
3. To be specified by each iSBX MULTIMODULE board.

* For a more complete definition of symbols refer to iSBX Bus Specification, 142686-001.

APPENDIX C
LISTING FOR THE iSBX™ DESIGN EXAMPLE SOFTWARE EXERCISER

```
LOC  OBJ          LINE        SOURCE STATEMENT

                     1 ;*****************************************************************************
                     2 ;*                                                                           *
                     3 ;*       THIS PROGRAM WAS USED AS AN EXAMPLE FOR EXERCISING THE              *
                     4 ;*       8279 iSBX MULTIMODULE BUILT FOR THIS APPLICATION NOTE.              *
                     5 ;*                                                                           *
                     6 ;*****************************************************************************
                     7
                     8
                     9 ;*****************************************************************************
                    10 ;                        PROGRAM EQUATES
                    11 ;*****************************************************************************
                    12
00F0                13 DATAAD           EQU             0F0H    ; PORT ADDRESS TO READ OR WRITE
                    14                                          ;/DATA TO/OR FROM KEYBOARD/DISPLAY
00F1                15 CMDAD            EQU             0F1H    ; PORT ADDRESS TO SEND COMMANDS
                    16                                          ;/TO KEYBOARD/DISPLAY
0008                17 MODE0            EQU             08H     ; CONTROL CHAR. TO SET
                    18                                          ;/KEYBOARD/DISPLAY MODE FOR
                    19                                          ;/(2 KEY LOCKOUT,16 CHAR LEFT ENTRY
0039                20 PROGCK           EQU             39H     ; CONTROL CHAR. TO SET 8279 CLK
                    21                                          ;/TO 100 KHz INTERNAL TIMING
0040                22 RDFIFO           EQU             40H     ; CONTROL CHAR. TO READ KEYBOARD
0060                23 RDRAM            EQU             60H     ; CONTROL CHAR. TO READ DISPLAY RAM
0070                24 RDRAMA           EQU             70H     ; CONTROL CHAR. TO READ DISPLAY RAM
                    25                                          ;/AUTO INCREMENT
0080                26 WRRAM            EQU             80H     ; CONTL CHAR. TO WRITE TO DISPLAY RAM
0090                27 WRRAMA           EQU             90H     ; CONTL CHAR. TO WRITE TO DISPLAY
                    28                                          ;/RAM AUTO INCREMENT
00D8                29 CLR              EQU             0D8H    ; CONTROL CHAR. TO CLEAR OR BLANK
                    30                                          ;/DISPLAY
3C00                31 BUFF             EQU             3C00H   ; ADDRESS OF KEYBOARD INPUT BUFFER
3D00                32 DBUFF            EQU             3D00H   ; ADDRESS OF DISPLAY BUFFER
                    33
                    34 ;*****************************************************************************
                    35
0000 F3             36 START:           DI
0001 C33B00         37                  JMP     BEGIN
                    38
                    39 ;************************   RST 7 ENTRY POINT   *********************************
                    40
0038                41                  ORG     38H
0038 C3D100         42                  JMP     INT
                    43
                    44 ;*****************************************************************************
                    45 ;                        INITIALIZE PROGRAM
                    46 ;                   AND KEY BOARD DISPLAY CONTROLLER
                    47 ;
003B 31FF3F         48 BEGIN:           LXI     SP,3FFFH        ; INITIALIZE STACK PT
003E 3E08           49                  MVI     A,MODE0         ; GET CONTROL CHAR.
0040 D3F1           50                  OUT     CMDAD           ; SET KEYBOARD/DISPLAY MODE
0042 3E39           51                  MVI     A,PROGCK        ; GET CONTROL CHAR.
0044 D3F1           52                  OUT     CMDAD           ; SET 8279 CLK FOR 100 KHz
0046 3ED8           53                  MVI     A,CLR           ; GET CONTROL CHAR.
0048 D3F1           54                  OUT     CMDAD           ; CLEAR OR BLANK DISPLAY
004A 0EE0           55                  MVI     C,0E0H
004C 21003C         56                  LXI     H,BUFF          ; SET POINTER TO INPUT BUFFER
004F 71             57                  MOV     M,C             ; CLEAR INPUT BUFFER TO BLANK CODE
0050 060F           58                  MVI     B,0FH           ; SET COUNTER = 15
0052 210F3D         59                  LXI     H,DBUFF+0FH     ; SET POINTER TO DBUFF +15
0055 71             60 ZDBUFF:          MOV     M,C             ; CLEAR DISPLAY BUFFER TO
0056 2B             61                  DCX     H               ;/DISPLAY BUFFER +15 TO CODE
0057 05             62                  DCR     B               ;/FOR CLEARING OR BLANKING OUT
0058 C25500         63                  JNZ     ZDBUFF          ;/THE DISPLAY
                    64
                    65 ;*****************************************************************************
                    66 ;                  THIS IS THE BACKGROUND PROGRAM
                    67 ;              WHICH LOOPS CHECKING FOR THE DATA PRESENT FLAG
                    68 ;
005B F3             69 CKFLAG:          DI                      ; DISABLE INTERRUPTS
005C AF             70                  XRA     A               ;/CLEAR A REG AND COMPARE WITH
005D B9             71                  CMP     C               ;/C REG CHECKING FOR DATA PRESENT
005E CA6400         72                  JZ      LABEL           ;/IF PRESENT CALL OUTPT
0061 CD6800         73                  CALL    OUTPT           ;/TO DISPLAY CHAR.
0064 FB             74 LABEL:           EI                      ;/IF NO DATA PRESENT ENABLE
0065 C35B00         75                  JMP     CKFLAG          ;/INTERRUPTS AND JMP BACK
                    76
```

```
                        ;*****************************************************************************
                        ;                       OUTPUT CHARACTER TO DISPLAY
                        ;
0068 3A003C             OUTPT:          LDA     BUFF            ; LOAD A WITH KEYBOARD DATA
006B 062B                               MVI     B,2BH           ; SET COUNTER MAX POSSIBLE CHAR.
006D 21DE00                             LXI     H,TABLE1        ; SET POINTER TO INPUT TABLE
0070 110901                             LXI     D,TABLE2        ; SET POINTER TO OUTPUT TABLE
0073 BE                 COMPARE:        CMP     M               ; COMPARE KEYBD DATA TO INPUT
0074 CA8000                             JZ      MATCH           ;/TABLE IF = JMP TO MATCH
0077 05                                 DCR     B               ;/ELSE DECREMENT COUNTER IF 0
0078 CAC600                             JZ      CONTROL         ;/JMP TO CONTROL
007B 23                                 INX     H               ;/ELSE INCREMENT BOTH TABLE
007C 13                                 INX     D               ;/POINTERS AND JMP TO COMPARE
007D C37300                             JMP     COMPARE
0080 EB                 MATCH:          XCHG                    ; IF MATCH CHANGE INPUT WITH
0081 7E                                 MOV     A,M             ;/OUTPT DATA AND PLACE IN BUFF
0082 21003C                             LXI     H,BUFF
0085 77                                 MOV     M,A
0086 060F                               MVI     B,0FH           ; SET COUNTER = TO 15
0088 11003D                             LXI     D,DBUFF         ; POINTER TO FIRST LOC IN DBUFF
008B 21013D                             LXI     H,DBUFF+1       ; POINTER TO 2ND LOC IN DBUFF
008E 7E                 LOOP1:          MOV     A,M             ; READ HIGH POINTER FROM DBUFF
008F 23                                 INX     H               ;/UPDATE HIGH POINTER
0090 EB                                 XCHG
0091 77                                 MOV     M,A             ; SHIFT DATA LEFT IN D BUFF
0092 23                                 INX     H               ; UPDATE LOW POINTER
0093 EB                                 XCHG
0094 05                                 DCR     B               ; TEST IF DONE
0095 C28E00                             JNZ     LOOP1           ;/AND GO BACK IF NOT
0098 3A003C                             LDA     BUFF            ;/ELSE READ KEYBOARD DATA
009B 320F3D                             STA     DBUFF+0FH       ;/AND PLACE IT IN THE DBUFF
009E 0610               LOOPA:          MVI     B,10H           ; SET COUNTER = 16
00A0 21003D                             LXI     H,DBUFF         ; SET POINTER = DBUFF 1ST POS.
00A3 7E                 LOOP2:          MOV     A,M             ;/READ 1 BYTE FROM DBUFF
00A4 D3F0                               OUT     DATAAD          ;/AND SENT IT TO DISPLAY
00A6 23                                 INX     H               ; UPDATE POINTER
00A7 05                                 DCR     B               ;/AND TEST IF DONE
00A8 C2A300                             JNZ     LOOP2           ;/GO BACK IF NOT DONE
00AB 0E00                               MVI     C,0H            ;/ELSE CLR DATA PRESENT FLAG
00AD C9                                 RET                     ;/AND RETURN
                        ;*****************************************************************************
                        ;                           CHARACTER DELETE
                        ;                             OR RUB OUT
                        ;
00AE 060F               DELETE:         MVI     B,0FH           ; SET COUNTER =15
00B0 110F3D                             LXI     D,DBUFF+0FH     ; SET POINTER = DBUFF+15
00B3 210E3D                             LXI     H,DBUFF+0EH     ; SET POINTER = DBUFF+14
00B6 7E                 LOOPB:          MOV     A,M             ; READ LOW POINTER FROM DBUFF
00B7 2B                                 DCX     H               ;/UPDATE LOW POINTER
00B8 EB                                 XCHG
00B9 77                                 MOV     M,A             ; SHIFT DATA RIGHT IN DBUFF
00BA 2B                                 DCX     H               ;/UPDATE HIGH POINTER
00BB EB                                 XCHG
00BC 05                                 DCR     B               ; TEST IF DONE
00BD C2B600                             JNZ     LOOPB           ;/AND GO BACK IF NOT
00C0 EB                                 XCHG                    ;/ELSE SET DBUFF FOR
00C1 36E0                               MVI     M,0E0H          ;/CODE TO BLANK DISPLAY
00C3 C39E00                             JMP     LOOPA           ;/AND JMP TO LOOPA
                        ;*****************************************************************************
                        ;                       CHECK IF CHARACTER IS
                        ;                   A CONTROL OR DELETE CHARACTER
                        ;
00C6 FEFA               CONTROL:        CPI     0FAH            ; COMPARE FOR CONTROL CHAR.
00C8 CA3B00                             JZ      BEGIN           ;/IF CONTROL JMP TO BEGIN
00CB FEF9                               CPI     0F9H            ;/ELSE COMP. FOR DELETE CHAR.
00CD CAAE00                             JZ      DELETE          ;/IF DELETE JMP TO DELETE
00D0 C9                                 RET                     ;/ELSE RETURN
                        ;*****************************************************************************
                        ;                         KEYBOARD INPUT
                        ;                       INTERRUPT ROUTINE
                        ;
00D1 3E40               INT:            MVI     A,RDFIFO        ; GET CONTL CHAR. TO READ FIFO
00D3 D3F1                               OUT     CMDAD           ; SET 8279 FOR READ MODE
00D5 DBF0                               IN      DATAAD          ; READ KEYBOARD DATA IN
00D7 21003C                             LXI     H,BUFF          ; SET POINTER TO BUFF
00DA 77                                 MOV     M,A             ;/AND STORE KEYBOARD DATA
00DB 0EFF                               MVI     C,0FFH          ;/THEN SET DATA PRESENT FLAG
00DD C9                                 RET                     ;/AND RETURN
```

```
                        158 ;*****************************************************************************
                        159 ;                                   TABLE 1
                        160 ;                ACCEPTABLE INPUT CHARACTERS FROM KEYBOARD
                        161 ;
00DE DE                 162 TABLE1:         DB      0DEH,0FFH,0EFH,0EEH,0E5H,0F6H,0FEH,0C6H
00DF FF
00E0 EF
00E1 EE
00E2 E5
00E3 F6
00E4 FE
00E5 C6
00E6 C9                 163                 DB      0C9H,0CAH,0D2H,0DAH,0D3H,0C7H,0D1H,0D9H
00E7 CA
00E8 D2
00E9 DA
00EA D3
00EB C7
00EC D1
00ED D9
00EE D5                 164                 DB      0D5H,0EDH,0E6H,0F5H,0C1H,0F7H,0DDH,0E7H
00EF ED
00F0 E6
00F1 F5
00F2 C1
00F3 F7
00F4 DD
00F5 E7
00F6 FD                 165                 DB      0FDH,0DFH,0CCH,0D4H,0DCH,0E4H,0ECH,0F4H
00F7 DF
00F8 CC
00F9 D4
00FA DC
00FB E4
00FC EC
00FD F4
00FE FC                 166                 DB      0FCH,0C0H,0C8H,0D0H,098H,0A2H,0CFH,0AAH
00FF C0
0100 C8
0101 D0
0102 98
0103 A2
0104 CF
0105 AA
0106 EB                 167                 DB      0EBH,0E3H,0D8H
0107 E3
0108 D8
                        168
```

```
                    169 ;*******************************************************************************
                    170 ;                                   TABLE 2
                    171 ;                   ACCEPTABLE OUTPUT CHARACTERS TO DISPLAY
                    172 ;
  0109 C1           173 TABLE2:         DB      0C1H,0C2H,0C3H,0C4H,0C5H,0C6H,0C7H,0C8H
  010A C2
  010B C3
  010C C4
  010D C5
  010E C6
  010F C7
  0110 C8
  0111 C9           174                 DB      0C9H,0CAH,0CBH,0CCH,0CDH,0CEH,0CFH,0D0H
  0112 CA
  0113 CB
  0114 CC
  0115 CD
  0116 CE
  0117 CF
  0118 D0
  0119 D1           175                 DB      0D1H,0D2H,0D3H,0D4H,0D5H,0D6H,0D7H,0D8H
  011A D2
  011B D3
  011C D4
  011D D5
  011E D6
  011F D7
  0120 D8
  0121 D9           176                 DB      0D9H,0DAH,0F1H,0F2H,0F3H,0F4H,0F5H,0F6H
  0122 DA
  0123 F1
  0124 F2
  0125 F3
  0126 F4
  0127 F5
  0128 F6
  0129 F7           177                 DB      0F7H,0F8H,0F9H,0F0H,0FDH,0EBH,0E0H,0EAH
  012A F8
  012B F9
  012C F0
  012D FD
  012E EB
  012F E0
  0130 EA
  0131 EF           178                 DB      0EFH,0EEH,02DH
  0132 EE
  0133 2D
  0000              179                 END     START

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
BEGIN   A 003B    BUFF    A 3C00    CKFLAG A 005B    CLR     A 00D8    CMDAD   A 00F1    COMPAR A 0073    CONTRO A 00C6
DATAAD A 00F0    DBUFF   A 3D00    DELETE A 00AE    INT     A 00D1    LABEL   A 0064    LOOP1  A 008E    LOOP2  A 00A3
LOOPA  A 009E    LOOPB   A 0086    MATCH  A 0080    MODE0   A 0008    OUTPT   A 0068    PROGCK A 0039    RDFIFO A 0040
RDRAM  A 0060    RDRAMA A 0070    START  A 0000    TABLE1 A 00DE    TABLE2 A 0109    WRRAM  A 0080    WRRAMA A 0090
ZDBUFF A 0055

ASSEMBLY COMPLETE,   NO ERRORS
```